CAAI Transactions on Intelligence Technology

Research Article

Teaching a robot to use electric tools with regrasp planning

ISSN 2468-2322
Received on 28th November 2018
Revised on 8th January 2019
Accepted on 9th January 2019
doi: 10.1049/trit.2018.1062
www.ietdl.org

Mohamed Raessa[1], Daniel Sánchez[1], Weiwei Wan[1,2] ✉, Damien Petit[1], Kensuke Harada[1,2]

[1]*School of Engineering Science, Osaka University, Osaka, Japan*
[2]*National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan*
✉ *E-mail: wan@sys.es.osaka-u.ac.jp*

Abstract: This study presents a straightforward method to teach robots to use tools. Teaching robots is crucial in quickly deploying and reconfiguring robots in next-generation factories. Conventional methods require third-party systems like wearable devices or complicated vision system to capture, analyse, and map human grasps, motion, and tool poses to robots. These systems assume lots of experience from their users. Unlike the conventional methods, this study does not involve learning human motion and skills. Instead, it only learns the object goal poses from the human user whilst employs regrasp planning to generate robot motion. The method is most suitable for a robot to learn the usage of electric tools that can be operated by simply switching on and off. The proposed method is validated using a dual-arm robot with hand-mounted cameras and several tools. Experimental results show that the proposed method is robust, feasible, and simple to teach robots. It can find a collision-free and kino-dynamic feasible grasp sequences and motion trajectories when the goal pose is reachable. The method allows the robot to automatically choose placements or handover considering the surrounding environment as intermediate states to change the pose of the tool and use tools following human demonstrations.

1 Introduction

Modularised manufacturing cells provide a promising way for the improvement of productivity in the manufacturing process. Various tools and workpieces are prepared in a manufacturing cell for a human worker to perform picking and assembly tasks. The transition to manufacturing cells proves to be efficient for reasons such as reducing the time required for transporting materials, preventing overproduction, as well as simplifying the production flow. In order for robots to take the place of human workers (Fig. 1) in manufacturing cells, several challenges have to be overcome such as teaching the robots which workpieces and tools to use for a given process, how to pick up workpieces and operate tools, and when to take advantages of the surrounding fixtures, and so on.

This research presents a method to handle one of the stated problems in replacing human workers in a manufacturing cell, namely teaching a dual-arm robot to use electric tools using regrasp planning and visual recognition. The advantage of this method is that it does not require any third-party assisting systems like wearable devices or complicated visual tracking technology to capture, analyse, and map human grasps, motion, and tool poses to robots. Instead, the robot learns the starting pose and goal poses of an electric tool from human demonstration and automatically generates manipulation motion using a regrasp planner [1].

Multiple approaches have been developed to simplify the teaching of robots, such as learning from demonstration (LfD) [2], reinforcement learning (RL) [2, 3], and end-to-end learning [4, 5]. These approaches usually require some experience with third-party devices for a human to lead the teaching process. Each of the approaches has its own limitations. In detail, LfD uses the mapping from a human teacher to the robot learner to make use of the teaching data. Most LfD systems use third-party devices such as head mounted devices and other forms of motion capture tools to collect the demonstrated task data and employ intricate algorithms to collect sensory data and shadowing it. RL enables a robot to discover the near-optimal policy of doing tasks through trial and error with its environment. It desires much effort to experiment with real-world robotic systems and requires the definition of a cost function that can drive the system to converge and succeed [6]. Inverse RL, as a side method of RL, aims at solving for the reward function by assuming the knowledge of the policy for doing a task, which requires collecting all the states and actions pairs from human demonstrators. End-to-end learning directly predicts motor commands using sensor input. It requires large amounts of sensor data to train deep neural networks.

Unlike the conventional approaches, this paper proposes a straightforward method to teach robots to use tools by demonstration. The method is most suitable for a robot to learn the usage of electric tools that can be operated by simply switching on and off without complicated skills. During teaching, a human user poses the electric tools to the goal to show how to use it. The robot learns the object's goal poses (position and orientation) from the human user and employs regrasp planning to generate robot motion considering environmental constraints. The robot is not instructed to follow any human motion or task requirements. Robot motion sequences and trajectories are determined and generated by automatic motion planning.

The proposed method is validated by teaching a dual-arm robot to use multiple electric tools commonly seen in manufacturing cells. The tools include an electric screwdriver, an electric plug, an electric saw, an electric vacuum cleaner, and a mechanical workpiece (as an example to simulate the components used in assembly tasks). For teaching the goal poses of the tools, a human user is asked to demonstrate their usage by placing the tooltips against the targets (Fig. 2*a*). The dual-arm robot uses one of its hand cameras to recognise the tools and the goal poses. After teaching, the tools are placed on a work table in front of the robot at arbitrary poses, the robot detects the poses and uses them as the starting poses of regrasp planning. Both the start and the goal poses are visualised to the user through a graphical user interface for easy supervision (Fig. 2*b*). During the planning process, the regrasp planner generates a series of motion sequences and trajectories to reorient the tools to the goal poses taught by the

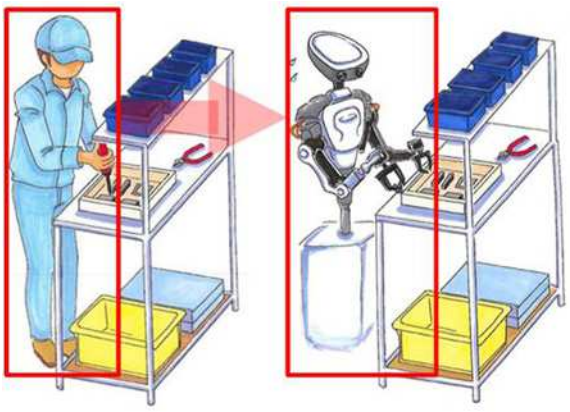

Fig. 1 *Replacing human workers in a manufacturing cell with dual-arm robots*

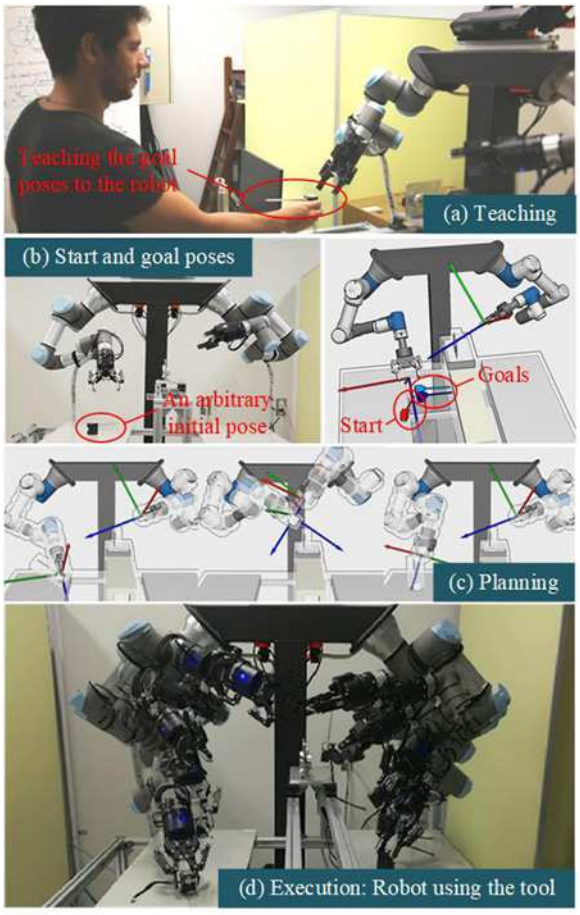

Fig. 2 *Teaching a robot to use a tool with regrasp planning*

a Human user teaches the robot the goal poses by demonstrating the usage of a tool
b After teaching, the tool is placed arbitrarily on the work table in front of the robot
c Arbitrary starting pose and the taught goal poses are used to plan regrasp motion sequences and trajectories to manipulate the tool
d Execution: the robot uses the tool following the planned motion

human user (Figs. 2*c* and *d*). The results of the various teaching show that the proposed system enables users with no related experience to intuitively teach the robot how to use different tools.

The rest of this paper is organised as follows. Section 2 reviews the work related to our study. Section 4 discusses the regrasp planning used for the object reorientation, and explains the system integration. Section 5 details the methods of marker-based visual recognition. Section 6 presents the experiments and shows their performance. Section 6.3 carries out a further discussion about the proposed system performance and highlights its promising points and challenges. Section 7 concludes our study.

2 Related work

2.1 Online and offline programming

This work studies teaching robots to use tools. Conventional methods can be divided into two categories, namely online and offline programming.

Online programming means collecting the required robot poses for a specific task by following an operator's instructions. The consecutive poses are recorded by a controller and are used to trace the same path repeatedly for the task. Recent studies in online programming showed an online humanoid robot motion generation application [7], the use of human motions for programming a robot [8], an online continuous human action recognition algorithm [9], and so on. Despite the advantages, online programming is limited to the operator's experience, and cannot adapt automatically to the changes in the robot's environment.

Offline programming depends on the full knowledge of the work cell and is more suitable to program the robots for complex tasks. Recent works related to offline programming include the implementation of interfaces for industrial robot's offline programming [10], the use of universal kriging to calibrate offline-programming industrial robots [11], a CAD-based offline solution for robot programming [12], and the applications of process planning and offline programming for robotic remote laser welding systems [13]. Offline programming can adapt in a better way to the changes of the grasped objects and can be enhanced for a more robust performance by using different sensory and vision systems [14, 15].

More specifically, several popular implementations of online and offline programming are available. They include LfD, RL, automatic motion planning, and so on. These implementations could be carried out both online and offline.

2.1.1 Learning from demonstration: LfD is an approach to generate robot motion by learning from demonstrated data. There are various approaches to collect the demonstration data from a teacher, who might be a human, another robot, or the robot itself. Third-party devices like motion capture systems or sensor data recording algorithms are widely used in LfD systems [2].

2.1.2 Reinforcement learning: RL, on the other hand, lets the robot autonomously discover the optimal behaviour for a task by interacting with the surrounding environment through trial and error. It has recently been applied to many areas including learning in robotics. Some limitations of RL include the difficulty in properly defining the reward functions and the high cost of trial and error in the real world or using dynamic simulations. Inverse RL goes the other way by assuming a known optimum behaviour and solves for the reward function by assuming the knowledge of the policy for doing a task [6].

2.1.3 Automatic motion planning: Automatic motion planning finds a collision-free trajectory for a robot to move objects from starting positions to goal positions. The popular motion planning algorithms include the optimisation-based methods like [16, 17] and probabilistic sampling-based method like [18, 19].

The goal of this study is to teach a robot without much human interference. Instead of learning human motion, we use motion planning to automatically generate robot motion. Thus, we are more interested in the literature related to automatic motion planning. Especially, we are interested in automatically planning motion sequences to reorient the pose of target objects (or tools). The research related to this topic is further reviewed below.

2.2 Reorientation planning

Industrial manipulators are usually equipped with simple grippers that are robust, easy to control, and cost-effective [20]. Those grippers are not dexterous enough for the manipulation tasks. Therefore, some techniques like task and context-sensitive optimisation for gripper design [21], vision-based grasping [22] and regrasping [23] have been studied and implemented. We are specially interested in regrasp planning which consists in finding a series of stable placements in order to reach the goal pose of an object. The object is moved between those placements by the means of shared grasps. This approach has three main steps: grasp planning in which the possible grasps are collected, placement planning in which the stable placements of the object are explored, and manipulation planning that usually builds a graph of those placements and their grasps. The task is done by specifying a path through this graph by selecting robot poses, and grasping configurations required to reach the final goal of the object [24, 25]. Another approach [26] presents the idea that the gripper dexterity does not have to be dependent only on the intrinsic resources of the gripper itself, but it can also be enhanced by the environment if used in a determined way, namely extrinsic manipulation. Some recent studies explore the idea of separation between the hands made to mimic the human hand and dexterous hands [27, 28]. Twelve reorientation actions were presented in [29]. A method of planning extrinsic manipulation based on motion-cones was presented in [30]. A visually controlled methodology to control the pivoting motion of extrinsic manipulation was presented in [31].

The robot motion sequences and trajectories in this paper are planned using regrasp planning. The basic algorithm is based on our previous work in [1, 24, 32]. The starting and goal configurations used by the regrasp planner are obtained using visual detection.

2.3 Visual detection

A method for recognising and tracking the pose of grasped objects is crucial to robotic manipulation. Various visual recognition and tracking techniques have been developed and applied to robotics such as detecting the line features [33], corners [34], or using scale-invariant features [35] using RGB images, 3D matching [36–38] using point clouds obtained from stereo cameras [39] and structured light patterns [40], and so on. Augmented reality (AR) markers, particularly, are used to assign digital information to items in the real world [41]. They are widely used in different fields [42–44] to provide pose information [45, 46]. The merits of using AR markers over the features extracted from the appearance are that they are easier to detect, and they can provide more precisely estimated poses [47].

In this research, we attach AR markers to electric tools to facilitate their detection and tracking. Attaching AR markers to the tools is a reasonable solution since the tools are deployed repeatedly in manufacturing cells. The advantages of the markers help the robot quickly and precisely find the tool in a manufacturing cell as well as learn its operation from demonstrators.

By integrating regrasp planning and AR markers, we implement a straightforward method to teach robots to use tools by demonstration. Compared with previous online and offline programming studies, our method does not require third-party systems like wearable devices nor complicated visual tracking technology, and does not assume professional experience from the demonstrators. Our method only involves learning goal poses from a human demonstrator and refers to regrasp planning to generate robot motions to reach the goal poses whilst considering environmental constraints. The method is particularly promising for a robot to learn the usage of electric tools that can be controlled by logic relays.

3 Overview of the system

Fig. 3 shows the workflow of the proposed direct teaching system. The workflow includes a teaching phase and a planning phase. In the teaching phase, a human demonstrator is asked to demonstrate where to move an electric tool to and in what pose should a robot use it. The human may move the tool to different positions and the system will record both the times and sequence of usage. In the planning phase, the planner plans a robot motion to move the tool from the start to the goal. The planner uses regrasp and handover to make the robot be able to reorient the pose of the tools.

The left green box in Fig. 3 indicates the starting pose learned from a human demonstrator. Here, the AR markers attached to the tool are detected in order to determine the tool's pose. A grasp database is used to retrieve candidate grasps to pick up the tool from the starting pose. This green box corresponds to the human teaching action in Fig. 2*a*. The robot uses a camera mounted on its waist to do pose detection. The demonstrator moves the tool to two poses. They are recognised by the robot and visualised as blue goals in Fig. 2*c*. The operation sequence is encoded by the different colour intensity – the darker blue indicates an early operation. The lighter blue indicates a later operation.

The right green box in Fig. 3 indicates the regrasp planning used to plan the motion sequences and motion trajectories. The input to the planner includes the observed starting pose and the taught goal poses. The planner will divide the planning into sub-tasks and plan the motion between them (e.g. the planner will plan the motion between the starting pose and the first goal pose, the first goal pose and the second goal pose etc., sequentially). At the end of each sub-task, the tool is moved to a goal pose and the robot will switch it on to do the jobs. The regrasp planner uses the grasps, the placements, and the handover configurations saved in the database to build regrasp graphs and compute the solutions. The planned results will be sent to real robots for execution.

Below we present in detail the implementations of the two components related to the two green boxes, namely the regrasp planning component and the vision component.

4 Regrasp planning component

Our regrasp planner was initially developed in [24]. It uses relational database [48] and regrasp graph [49] to save and plan different levels

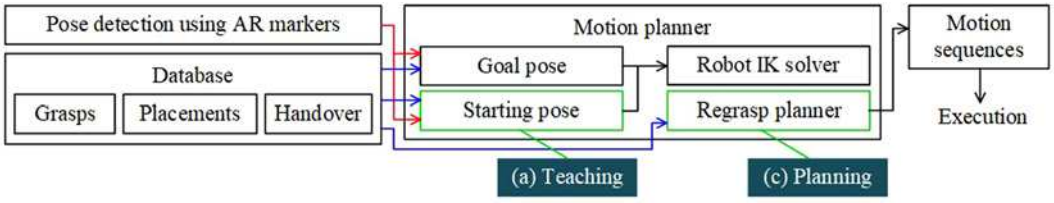

Fig. 3 *Framework of the work flow of the proposed direct teaching system. The work flow includes a teaching phase and a planning phase. In the teaching phase, a human user is asked to demonstrate where to move an electric tool to and in what pose should a robot use it. The human may move the tool to different positions and the system will record both the times and sequence of usage. In the planning phase, the planner plans a robot motion to move the tool from the start to the goal. The planner uses regrasp and handover to make the robot be able to reorient the pose of the tools*

CAAI Trans. Intell. Technol., 2019, Vol. 4, Iss. 1, pp. 54–63

of regrasp sequences for object reorientation. The regrasp planner accepts a starting pose and a goal pose as input parameters and returns a series of robot key poses and grasp configurations. The key poses are further used as the starting configuration and goal configuration of a motion planner to generate collision-free and IK-feasible robot motion.

4.1 Regrasp planning

The regrasp planning problem is a multi-modal searching problem through a bunch of high-dimensional configuration spaces connected by low-dimensional manifolds. The starting pose and goal pose of a regrasp planning problem are usually located in two different high-dimensional configuration spaces. The planned path has to pass through several low-dimensional manifolds to reach the goal. Fig. 4 shows an example of multi-modal planning. Formally, the search space of a regrasp planning problem is defined as follows:

$$\mathcal{S} = \{\mathcal{C}_0, \mathcal{M}_{01}, \mathcal{C}_1, \mathcal{M}_{12}, \mathcal{C}_2, \ \dots\}, \mathcal{M}_{ij}\mathcal{C}_i \wedge \mathcal{M}_{ij}\mathcal{C}_j \tag{1}$$

Here, $\mathcal{C}_0, \mathcal{C}_1, \dots$ denote the high-dimensional configuration spaces. $\mathcal{M}_{01}, \mathcal{M}_{12}, \dots$ denote the low-dimensional manifolds connecting the high-dimensional configuration space pairs $(\mathcal{C}_0, \mathcal{C}_1)$, $(\mathcal{C}_1, \mathcal{C}_2)$, …, $\mathcal{M}_{ij}$ is a manifold that is embedded simultaneously in $\mathcal{C}_i$ and $\mathcal{C}_j$. The starting pose and goal pose are usually represented by

$$\boldsymbol{q}_s \in \mathcal{C}_i, \quad \boldsymbol{q}_g \in \mathcal{C}_j, \quad i \neq j \tag{2}$$

The first step of the regrasp planning problem is to find a sequence of key poses $\boldsymbol{p}_{\{0:K\}}$, where

$$\begin{aligned} &\boldsymbol{p}_0 \leftarrow \boldsymbol{q}_s, \quad \boldsymbol{p}_K \leftarrow \boldsymbol{q}_g \\ &\boldsymbol{p}_i \in \mathcal{C}_j, \quad \boldsymbol{p}_{i+1} \in \mathcal{M}_{j(j+1)}, \quad \boldsymbol{p}_{i+2} \in \mathcal{C}_{j+1}, \quad i = 0, 3, \ \dots \end{aligned} \tag{3}$$

Then, adjacent key poses are sent to motion planners to find a smooth trajectory $\boldsymbol{q}_{\{0:T\}}$, where

$$\boldsymbol{q}_0 \leftarrow \boldsymbol{p}_i, \quad \boldsymbol{q}_T \leftarrow \boldsymbol{q}_{i+1}, \quad \boldsymbol{q}_k \in \mathcal{C}_j, \quad k = 0, 1, 2, \ \dots \tag{4}$$

The finally planned motion will be

$$\text{traj} = \boldsymbol{q}_{\{0:T_0\},\{T_0:T_1\},\dots,\{T_x:K\}} \tag{5}$$

where each section $\boldsymbol{q}_{\{T_i:T_{i+1}\}}$ is a trajectory. All sections imply a motion sequence.

4.2 Regrasp planning using a 6-DoF robot with a 2-f gripper

In the regrasp planning of a 6-DoF robot with a parallel gripper, $\mathcal{C}_i$ is an SE(3) space. Each of the joint angles will be treated as a dimension of $\mathcal{C}_i$. $\mathcal{M}_{ij}$ is an $\text{SE}(2) \times S^1$ space, where the three dimensions before the $\times$ symbol correspond to the contact position (R^2) and the rotation of the gripper (S^1). The additional dimension after the S^1 symbol corresponds to the rotation of a grasp around the axis connecting the two contact points on the two finger pads of the 2-f gripper. The opening distance of the gripper in $\mathcal{M}_{ij}$ is determined by the configurations in its host spaces $\mathcal{C}_i$ and $\mathcal{C}_j$. It is not a changeable parameter and is therefore not treated as a dimension.

In the implementation, the $\mathcal{C}_i$ and $\mathcal{M}_{ij}$ are further modularised by grasps. $\mathcal{C}_i$ is identified by the opening distance of a gripper. A configuration in $\mathcal{C}_i$ denotes a pose of a robot with the 2-f gripper installed at its tool centre point (TCP) opened to a specific distance. $\mathcal{M}_{ij}$ is discretised into a set of hand configurations which are described by the position and rotation the hand. The discretisation makes the planned trajectory section in $\mathcal{M}_{ij}$ discrete. A robot is able to release and regrasp an object to do reorientation. On the other hand, if $\mathcal{M}_{ij}$ was not discretised, the planned trajectory section in $\mathcal{M}_{ij}$ will be continuous and a robot will use techniques like in-hand manipulation [30, 50] or extrinsic manipulation [51, 52] to change object poses.

The implementation includes a grasp planning module, an intermediate state planning module, and a regrasp graph module. The grasp planning module plans the grasps by analysing the geometric meshes and contact of the object and the hand. The placement planning plans stable placements (intermediate state 1) of the object on a fixture in the environment (e.g. a table) and the handover poses (intermediate state 2) for transferring the object from one robot hand to another. The regrasp graph module uses the planned grasps and intermediate states to build a graph. The regrasp planner will search the graph to find the sequence of key poses. $\boldsymbol{g}_i$ is used to represent the grasp that modularises $\mathcal{C}_i$. A configuration $\boldsymbol{q}$ in $\mathcal{C}_i$ could be represented by $\boldsymbol{q}^{\boldsymbol{g}_i}$. The motion trajectory in $\mathcal{C}_i$ is therefore $\text{traj}_i = \boldsymbol{q}^{\boldsymbol{g}_i}_{\{T_i:T_{i+1}\}}$. The configurations that connect $\mathcal{C}_i$ and $\mathcal{C}_j$ in $\mathcal{M}_{ij}$ are represented by $\boldsymbol{q}^{\boldsymbol{g}_i}$ and $\boldsymbol{q}^{\boldsymbol{g}_j}$. $\boldsymbol{g}_i$ and $\boldsymbol{g}_j$ are connected to each other considering whether they modularise adjacent configuration spaces and manifolds. The regrasp graph holds the connections. The grasps are planned by the grasp planning module. The manifolds $\mathcal{M}_{ij}$ s are identified using the intermediate states.

Some details are as follows. For planning the grasps, the planner uses 3D mesh models of the tool and the gripper to generate a set of grasps which are free from the collision between the gripper and the object. The planner also computes the stable placements of the object on a horizontal surface to find the stable intermediate states on a table. The stability of the placement is computed using the distance between the projection of the centre of mass and the object convex hull. If this distance is smaller than a given threshold, the placement is considered unstable and discarded. Collisions between the object and the placement surface are also

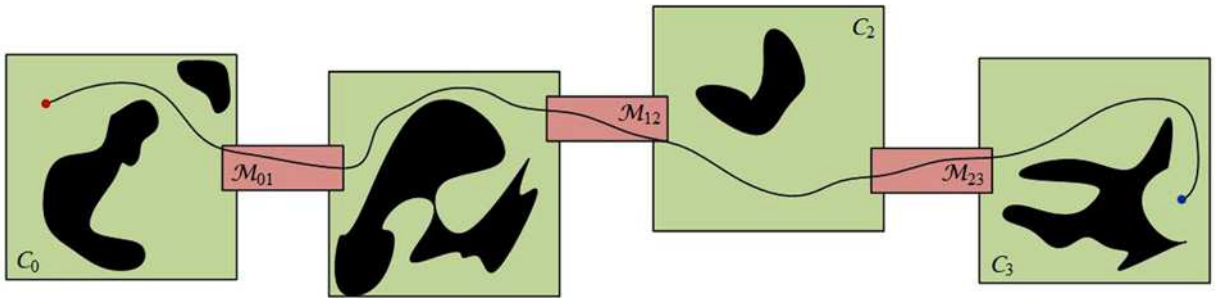

Fig. 4 *Regrasp planning problem is a multi-modal searching problem through a bunch of high-dimensional configuration spaces $\mathcal{C}_0, \mathcal{C}_1, \dots$ connected by low-dimensional manifolds $\mathcal{M}_{01}, \mathcal{M}_{12}, \dots$*

checked. Then, the set of grasps is refined by eliminating the grasps that collide with the horizontal placement surface. The other intermediate states, namely the handover and their associated grasps, are computed in the same way except that the stability is computed using the quality of grasp force closure [53]. The inverse kinematics for the robot are solved for stable placements and object handover, and the planner builds a graph of the states and their corresponding grasps. The graph is the aforementioned regrasp graph. The green net in Fig. 5*a* shows an example. During the regrasp task, the planner searches the regrasp graph to find a path of feasible placements and handover between the initial pose and the goal pose. The path is converted to the sequence of keyposes. The black segments in Fig. 5*a* shows the path. Fig. 5*b* shows the converted keyposes.

4.2.1 Motion planning: The adjacent configurations in the keypose sequence (e.g. Fig. 5*b*) are then sent to a motion planner to find a motion trajectory following (5). The planning also has two modules. One is the path planning module, the other one is the trajectory optimisation module. The path planning could be done using widely seen planners like PRM [54], RRT [55], or their variations (RRT-connect [19], Dynamic Domain RRT [56], RRT star [57] etc.). Each planner has its advantages. For example, the PRM method reuse pre-evaluated nodes and is considerably fast. The RRT and RRT-connect methods easily integrate with various constraints and output smooth motion. The transition-RRT method [58] and bridging methods [59] are good at navigating through narrow configuration passages. In our implementation, we do not fix the planner to a specific candidate. Instead, the path planning algorithms are designed to be replaceable. Users may determine their path planners following the requirements of their tasks. The path found by the path planner is smoothed by a random cut algorithm [60] and optimised using five-order polynomial to avoid jerk [61]. The final output will be a smooth robot trajectory with reasonable timestamps. A comparison of the planned path, the smoothed path, and the optimised trajectory is shown in Fig. 6.

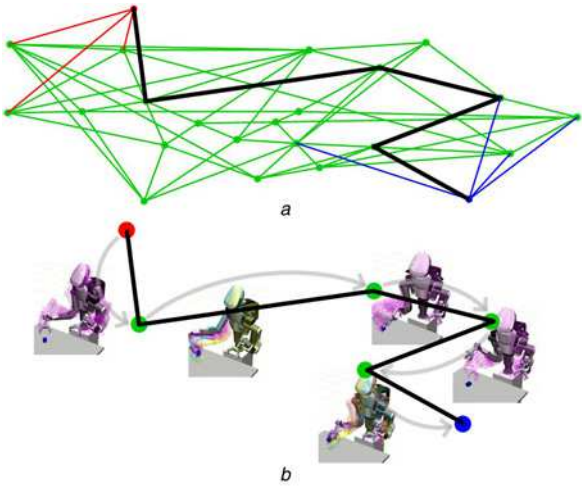

Fig. 5 *The regrasp graph and the planned robot key poses*

a A planner searches a regrasp graph built considering the intermediate states to get a path of feasible placements and handover between the initial pose and the goal pose
b the path of states are converted to a sequence of keyposes for motion planning

5 Vision component

The pose detection of the manufacturing cell tools is done through AR markers attached to the tool surface. A hand-mounted RGB camera captures the pose of the tool in the camera's local coordinate system. The local pose is then transformed to obtain the position and orientation of the tool in the world coordinate system. A visual interface is used to illustrate the position of the robot, its environment, and the tool positions, making the system intuitive and easy to use for human users. Using AR markers with the tools in the manufacturing cell can be a typical application in which the markers do not impose a limitation as the same work cell tools are to be used repeatedly. This section described the calibration of the hand-mounted cameras and how the object pose is obtained in the world frame at any arm pose.

5.1 Calibration of the hand-mounted cameras

Fig. 7 shows the problem definition. The relative pose of the marker in the camera's local coordinate system is defined as ${}^{\Sigma_C}\boldsymbol{T}_{\Sigma_M}$, where Σ_M indicates the pose of the marker (represented by a local coordinate system), Σ_C indicates the local coordinate system of the camera. For simplicity, we remove the Σ_C symbol in the transformation matrix and use ${}^{C}\boldsymbol{T}_{M}$ to represent ${}^{\Sigma_C}\boldsymbol{T}_{\Sigma_M}$ in the following text. Likewise, the relative pose of the camera to the world, namely the homogeneous transformation from the camera's local coordinate system to the world coordinate system, is defined as ${}^{W}\boldsymbol{T}_{C}$. The homogeneous transformation from the marker to the tool local coordinate system is defined as ${}^{O}\boldsymbol{T}_{M}$. The goal of calibration is to find the homogeneous transformation from the robot's TCP, say, the local coordinate system of the end effector, Σ_{EE} to Σ_C.

Using AR markers, we can find the poses of the AR markers in the camera coordinate system. The transformation from the marker to the world frame ${}^{W}\boldsymbol{T}_{M}$ is equivalent to the composed transformations from the marker to the camera ${}^{C}\boldsymbol{T}_{M}$ (detected using vision), and the transformation from the camera to the world coordinate system

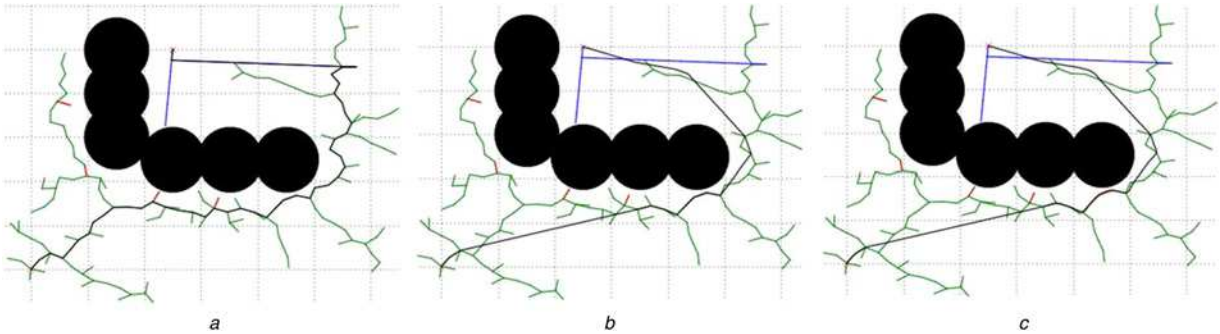

Fig. 6 *Comparison among different planning results*

a Planned path
b Smoothed path
c Optimised trajectory
When computing the smoothed path, the iteration times were set to 30. The maximum speed and control interval used to compute the optimised trajectory are 0.1 m/s and 0.008 s. The length of the grids in the figure is 0.1 m. The black regions are obstacles. The curves in (c) are slightly deviated from (b) due to dynamic constraints

CAAI Trans. Intell. Technol., 2019, Vol. 4, Iss. 1, pp. 54–63

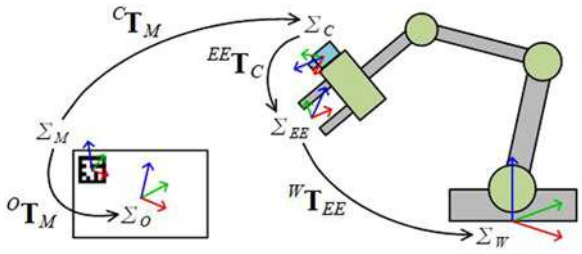

Fig. 7 *Various coordinate systems used to calibrate a hand-mounted camera. The camera is drawn as blue blocks in figure*

$^{W}\boldsymbol{T}_{C}$. Thus, the transformation from the camera to the world coordinate system can be computed according to (7)

$$^{W}\boldsymbol{T}_{M} = {}^{W}\boldsymbol{T}_{C}^{C}\boldsymbol{T}_{M} \tag{6}$$

$$^{W}\boldsymbol{T}_{C} = {}^{W}\boldsymbol{T}_{M}^{C}\boldsymbol{T}_{M}^{-1} \tag{7}$$

On the other hand, the transformation from the end effector to the world coordinate system $^{W}\boldsymbol{T}_{EE}$ can be precisely obtained using the kinematics of the robot. The transformation from the camera to the world coordinate system $^{W}T_{C}$ can be computed as follows:

$$^{W}\boldsymbol{T}_{C} = {}^{W}\boldsymbol{T}_{EE}^{EE}\boldsymbol{T}_{C} \tag{8}$$

The transformation from the camera to the end-effector coordinate system is therefore

$$^{EE}\boldsymbol{T}_{C} = {}^{W}\boldsymbol{T}_{EE}^{-1W}\boldsymbol{T}_{C} \tag{9}$$

By substituting the $^{W}\boldsymbol{T}_{C}$ in (9) using (7), $^{EE}T_{C}$ is expressed as

$$^{EE}\boldsymbol{T}_{C} = {}^{W}\boldsymbol{T}_{EE}^{-1W}\boldsymbol{T}_{M}^{C}\boldsymbol{T}_{M}^{-1} \tag{10}$$

As mentioned before, $^{W}\boldsymbol{T}_{EE}$ can be precisely obtained using the kinematics of the robot. $^{C}\boldsymbol{T}_{M}$ is the output of visual detection. $^{EE}\boldsymbol{T}_{C}$ can be computed by properly setting markers to some known positions in the world (to let $^{W}\boldsymbol{T}_{M}$ to be known).

The way we set the markers is using another robot manipulator. A marker is attached to the end-effector of the second manipulator. The coordinate system of the marker is manually set to be the same as the coordinate system of the end effector. The configuration is shown in Fig. 8. In this case, (11) is converted to

$$^{EE}\boldsymbol{T}_{C} = {}^{W}\boldsymbol{T}_{EE}^{-1W}\boldsymbol{T}_{EE_2}^{C}\boldsymbol{T}_{M}^{-1} \tag{11}$$

where $^{W}\boldsymbol{T}_{EE_2}$ denotes the transformation from the second end effector to the world coordinate system. It can be precisely obtained using the kinematics of the second robot. Up to this point, all values in the right part of (11) are known, and the $^{EE}\boldsymbol{T}_{C}$ can be solved. In practice, we further move the end effector of the second manipulator to different positions to get several $^{W}\boldsymbol{T}_{EE_2}$, and employ mean square error optimisation to get an optimal $^{EE}\boldsymbol{T}_{C}$.

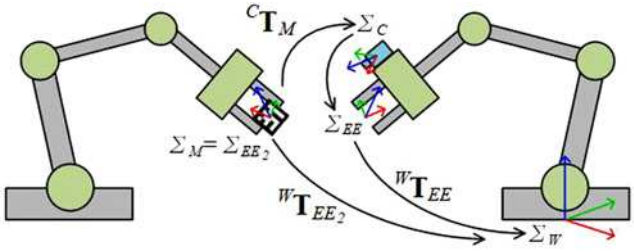

Fig. 8 *Using another robot manipulator to hold the AR marker for calibration. The coordinate system of the marker is manually set to be the same as the coordinate system of the end effector*

5.2 Detecting the pose of a tool

Next, the transformation of an object's local coordinate system to the world coordinate system, $^{W}\boldsymbol{T}_{O}$, could be decomposed into the multiplication of the following transformations:

$$^{W}\boldsymbol{T}_{O} = {}^{W}\boldsymbol{T}_{EE}^{EE}\boldsymbol{T}_{C}^{C}\boldsymbol{T}_{M}^{M}T_{O} \tag{12}$$

The equation exactly follows the arrows in Fig. 7. $^{W}\boldsymbol{T}_{O}$ is decomposed to the transformation from the tool to the marker

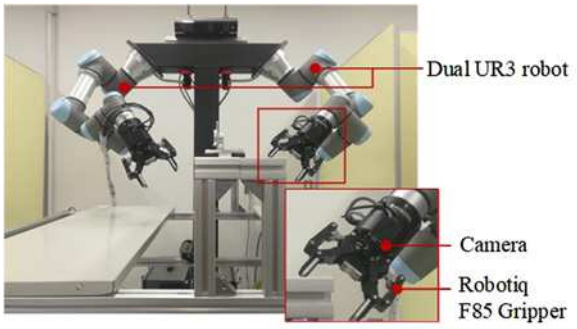

Fig. 9 *Experimental setup and the different components of the system. The setup simulates a manufacturing cell with dual UR3 robots. Each arm has a Robotiq F85 two finger grip. Cameras for marker detection are mounted at the palm of the gripper*

Fig. 10 *Tools used in the experiments*

a Electric screwdriver
b Electric plug
c L-shaped object from belt-drive unit
d Electric saw
e Mini vacuum cleaner

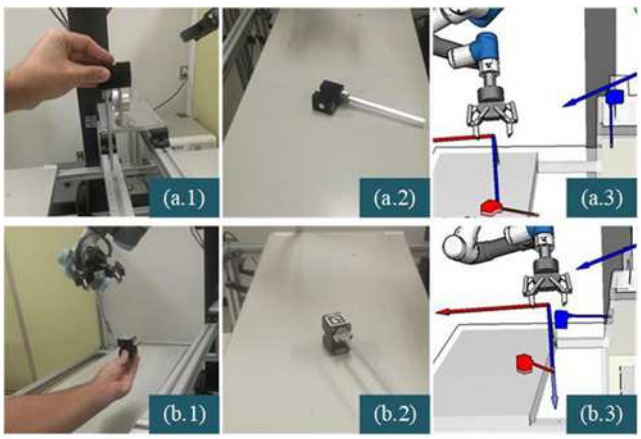

Fig. 11 *Teaching the robot to use an electric driver*

*a*1, *b*1 Taught goal poses
*a*2, *b*2 Starting poses
*a*3, *b*3 Recorded starting poses (red) and goal poses (blue) (best viewed in colour online)

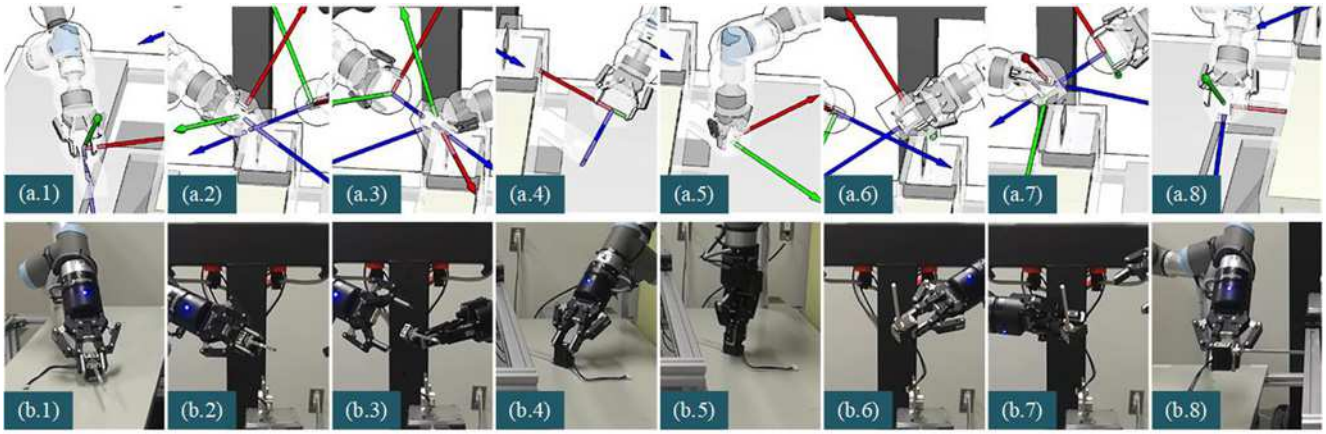

Fig. 12 *Results of learning to use the electric driver*

*a*1–*a*8 Planned motion
*b*1–*b*8 Real-world execution

${}^{M}T_{O}$, multiplied by the transformation from the marker to the camera ${}^{C}\boldsymbol{T}_{M}$, multiplied by the transformation from the camera to the end effector ${}^{EE}\boldsymbol{T}_{C}$, and finally multiplied by the transformation from the end effector to the world ${}^{W}\boldsymbol{T}_{EE}$.

These matrices are also known. ${}^{M}T_{O}$ is determined when the marker is attached to the tool. ${}^{C}\boldsymbol{T}_{M}$ is the output of visual detection. ${}^{EE}\boldsymbol{T}_{C}$ is calibrated using (11). ${}^{W}\boldsymbol{T}_{EE}$ be precisely obtained using the kinematics of the robot. The pose of a tool can be computed accordingly.

6 Experiments and analysis

6.1 Experimental setup

We use a dual UR3 [Universal Robots: https://www.universal-robots.com/products/ur3-robot/.] robot to validate the developed teaching system. The robot setup (the manufacturing cell) is shown in Fig. 9. Two UR3 arms are hanged with 45° on a support frame. Each arm has six DoFs. A Robotiq F85 two finger gripper [Robotiq: https://robotiq.com/products/adaptive-grippers.] and a Robotiq FT300 [Robotiq: https://robotiq.com/products/ft-300-force-torque-sensor.] force-and-torque sensor are installed to the end of the last link as the end effector. An ELP-USBFHD06H-L36 skewless HD camera [ELP: http://www.webcamerausb.com.] is mounted to one side of the Robotiq F85's palm for visual detection. The cameras have a resolution up to 1920×1080 pixels with 30 fps. The camera-gripper setup is zoomed up in the bottom-right corner of Fig. 9. The environment of the robot is set to simulate a manufacturing cell with a work table and some assembly parts in the work space. All the parts are considered for collision detection while planning the motion of the robot. The system is controlled using PyManipulator [PyManipulator: https://github.com/wanweiwei07/pymanipulator.].

6.2 Tasks and performance

In the experiments, the robot learns visually from a human how to use five tools shown in Fig. 10. The tools include an electric driver, a plug, and an L-shaped object from a belt-drive unit to represent a general assembly part, an electric saw, and a mini vacuum cleaner. Both teaching and automatic planning are performed for these tools.

6.2.1 Electric driver: In the first task, during the teaching, a human holds the electric screw driver in its desired goal pose. The robot learns how to pose the tool into the goal shown by the human. The goal pose is saved for the planner to use in the task mode. Figs. 11*a*1 and *b*1 show two teaching examples. The recorded goal poses are shown in blue in Figs. 11*a*3 and *b*3.

During planning and execution, the tool is put at a random starting pose on the work table. Figs. 11*a*2 and *b*2 show two examples. The starting pose is detected by the camera, and is used as the input of the regrasp planner together with the taught goal pose. The planner plans the required regrasps along with their corresponding robot poses and end effector configurations to reorient the tool from this start pose to the goal pose. The detected poses corresponding to Figs. 11*a*1 and *b*1 are shown in red in Figs. 11*a*3 and *b*3.

Figs. 12*a*1–*a*8 show the results of the regrasp planner. The robot employs two handover (Figs. 12*a*2, *a*3 and *a*6, *a*7) and one time of placement (Figs. 12*a*4 and *a*5) to reorient the tool. The real-world execution is shown in Figs. 12*b*1–*b*8.

6.2.2 Plug: In the second task, the robot learns about the goal pose of a plug from the human demonstrator. Similarly, Fig. 13 shows the two examples carried out with the electric plug. The starting and goal poses of the tool in each example are detected and saved. They are drawn in red blue in the simulation environment. A snapshot sequence of one of the real-world execution is illustrated in Figs. 14*a*–*h*. Two times of handover (Figs. 14*d*–*g*) are used to regrasp the plug and make the goal reachable.

6.2.3 Other tools: In the third task, the robot learns how to pose an L-shaped mechanical part to simulate an assembly process in the

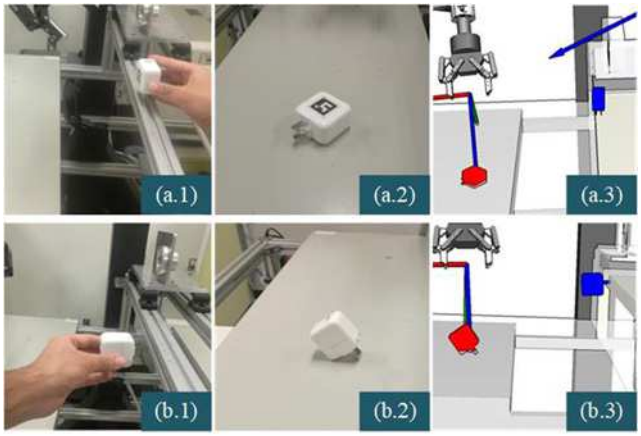

Fig. 13 *Teaching the robot to insert the power plug*

*a*1, *b*1Taught goal poses
*a*2, *b*2 Starting poses
*a*3, *b*3 Recorded starting poses (red) and goal poses (blue) (best viewed in colour online)

CAAI Trans. Intell. Technol., 2019, Vol. 4, Iss. 1, pp. 54–63

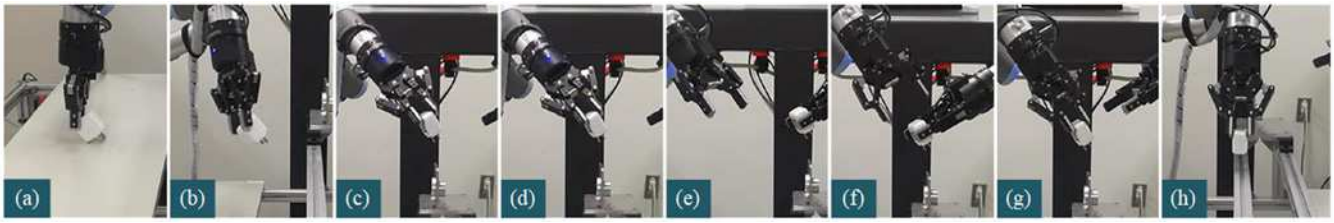

Fig. 14 *Executing the planned results to insert the power plug*

a–h Real-world execution of grasping the power plug from the detected start pose and posing it into the taught goal pose

work cell. The start pose, the taught goal pose, and their visualisation are shown in Figs. 15*a–c*. In the fourth and fifth tasks, the robot learns how to pose an electric saw, and a mini vacuum cleaner to goal poses that can be further used for completing another task using those tools. The starting poses and the taught goal poses of the saw and the vacuum cleaner as well as their visualisations in the simulation environment are shown in Figs. 16 and 17. The sequences of completing those two tasks in simulation and in reality are shown in Figs. 18 and 19, respectively. The robot is doing simple pick-and-place without using regrasp.

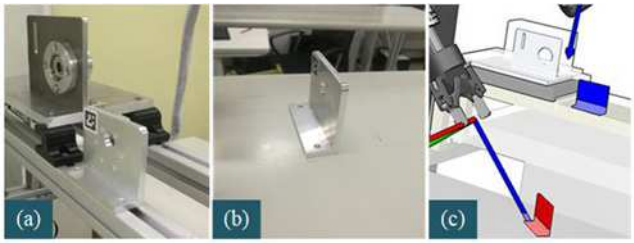

Fig. 15 *Teaching the robot to pose an L-shaped mechanical part*

a Taught goal poses
b Starting poses
c Recorded starting poses (red) and goal poses (blue) (best viewed in colour online)

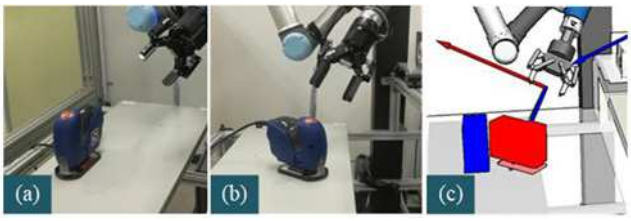

Fig. 16 *Teaching the robot to pose an electric saw*

a Taught goal poses
b Starting poses
c Recorded starting poses (red) and goal poses (blue) (best viewed in colour online)

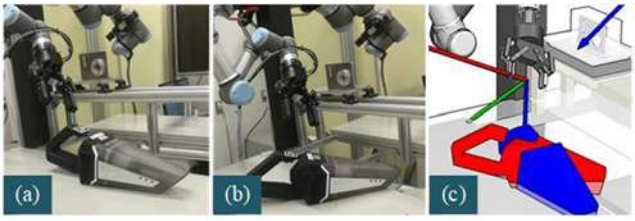

Fig. 17 *Teaching the robot to pose a mini vacuum cleaner*

a Taught goal poses
b Starting poses
c Recorded starting poses (red) and goal poses (blue) (best viewed in colour online)

6.2.4 Performance: All executions of these tools at the given poses are performed successfully during the experiments. The system can find a collision-free and kino-dynamic feasible grasp sequences and motion trajectories when the goal pose is reachable. The robot automatically chooses placements or handover as intermediate states to change the pose of the tool. The most costly result is the one shown in Fig. 12 where the robot employed two handover and one placement to regrasp and reorient the tool. The electric draw and the vacuum cleaner tasks are less expensive. They only require simple pick-and-place motion.

The regrasp planner does not necessarily find a successful motion. There are cases that the robot cannot reach the given goal, and the system reports failure. This usually happens to the electric draw

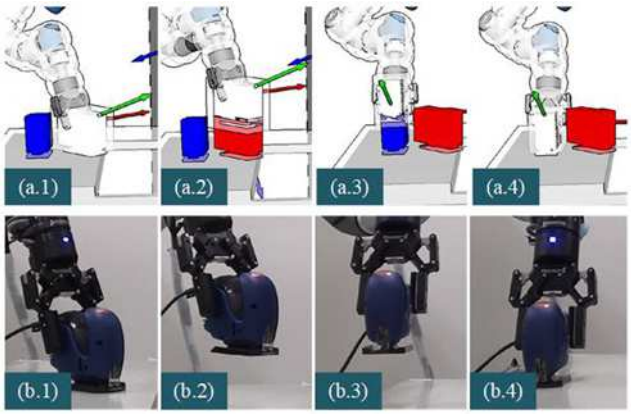

Fig. 18 *Executing the planned results to pose the saw*

*a*1–*a*4 Planned motion of grasping the saw from the detected starting pose (red) and posing it in the taught goal pose (blue) (best viewed in colour online)
*b*1–*b*4 Real-world executions

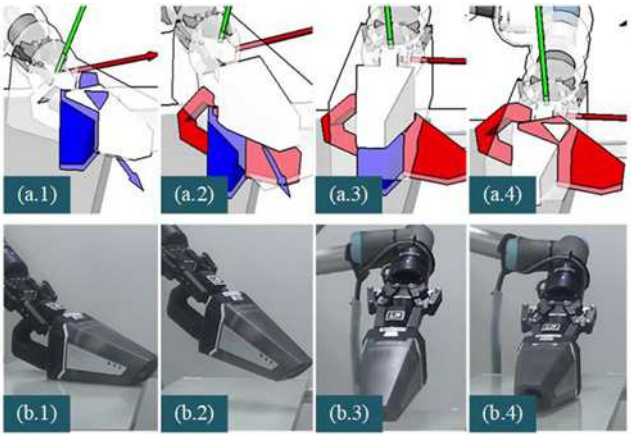

Fig. 19 *Executing the planned results to pose the vacuum cleaner*

*a*1–*a*4 Planned motion of grasping the vacuum cleaner from the detected starting pose (red) and posing it in the taught goal pose (blue) (best viewed in colour online)
*b*1–*b*4 Real-world executions

Table 1 Time costs of low-level motion planners

RRT	RRT-connect	Dynamic domain RRT	RRT star
0.016 s	0.026	0.175 s	3.253 s

and the vacuum cleaner. These two tools have large bodies with few graspable features. They are difficult to manipulate using 2-finger grippers. The difficulty is two-fold. For one thing, the regrasp graph has low connectivity due to the small number of available grasps. For the other, the motion planning may not be able to find a feasible motion due to the obstacles in the configuration space. The obstacles are formed by robot link and joint constraints, as well as objects in the environment.

The costs of different low-level motion planners (various RRT variations) are shown in Table 1. The results are the average values of ten single plans on an Intel Xeon E3-1505M v5 CPU. The results show that the RRT, RRT-connect, and Dynamic Domain RRT planner could find a path in <0.2 s. The reason is our target applications do not have difficult narrow passages. These RRT variations could meet the requirements of regrasp planners. An exception is the RRT star method. It is an optimisation method and is computationally intensive. Using the RRT star method as the low-level motion planners is inadvisable.

6.3 Discussion

The results show that the developed system is robust and can automatically choose intermediate states and plan robot motion. The teaching method is direct as the human user only need to demonstrate some goal poses. No intermediate motion is recorded and thus no special skills are desired from the demonstrators. The tools are assumed to be electric tools which can be switched on and off using an electric solenoid. These tools are simple to use – posing the tooltip to targets and turning on the power. There is no need to perform complicated skills like pushing, knocking, winding, and so on.

Also, the system allows teaching multiple goal poses. The human demonstrator may move the tooltip to different targets with a special task order. The system will record both the tool poses at these targets and their order. The planner is able to plan a motion to move the tool to the goal poses following the taught order.

The assumption of electric tools limits the application of the system. It cannot be used to teach the usage of general tools like manual drivers, wrenches, knives, and so on. In these cases, the robot not only needs to learn the goal pose of the tools, but also need to learn the motion trajectories. For example, the tooltip of a manual driver must be winded with a certain degree, released from the head of the screw, and winded back to start over. Using such tools needs complicated motion skills. There are two solutions to incorporate the motion skills. One is to save the motion skills as a skill database, and use affordance [62] and adaption [63] to retrieve and reuse the saved skills. The second one is to discretise the skills to some key goals and develops a planner that generates smooth motion to connect the key goals. The generated smooth motion is expected to approximate the taught skills and thus allow the robot to learn from human demonstration. The first solution has been explored by many far-seeing researchers in the last decades. It requires a large skill base and well-perceived target information for motion adaption, which is difficult to our system. The second one is a promising solution and is to be implemented as a future work. The implementation of teaching in this paper, as mentioned before, plans multiple regrasps between the taught goal poses. It is a preliminary implementation. Between the goal poses, the robot does not necessarily maintain the same grasping configuration. After reaching the previous goal, the robot may release the tool, regrasp it, and move it to the next goal. This implementation does not allow the robot to smoothly or continuously move to the taught goals (the discretised key goals) and generate smooth motion to approximate the complicated motion skills. In the future, we will develop a planner that allows maintaining the same grasp configuration for all discretised goals and enables teaching a robot to use non-electric tools.

7 Conclusions and future work

The paper introduced a direct method for teaching robots in manufacturing cells how to use electric tools. The proposed method integrates regrasp planning with vision feedback for detecting starting poses and generates robot motion to move the tools to the taught goal poses. Through multiple experiments, the proposed method proves robust, feasible, and simple to the teaching of robots. It can find a collision-free and kino-dynamic feasible grasp sequences and motion trajectories when the goal pose is reachable. The method allows the robot to automatically choose placements or handover considering the surrounding environment as intermediate states to change the pose of the tool.

To our best knowledge, the method is the first study that uses regrasp planning for LfD. Despite the novelty and feasibility, there is a long way to go to finally implement full direct teach and planning. The first problem is precision. While the tool could be manipulated to the goal pose, it cannot be easily attached to targets like the head of a screw, the holes of an electric outlet, and so on. It is necessary to include real-time visual feedback with the AR markers and force control to obtain the correct attachment. Second, as mentioned in the experimental section, it would be advisible to develop a planner that can maintain the same grasp configurations for all taught goals and enables teaching a robot complicated motion skills.

8 References

[1] Wan, W., Harada, K.: 'Developing and comparing single-arm and dual-arm regrasp', *IEEE Robot. Autom. Lett.*, 2016, pp. 243–250

[2] Argall, B.D., Chernova, S., Veloso, M., *et al.*: 'A survey of robot learning from demonstration', *Robot. Auton. Syst.*, 2009, **57**, (5), pp. 469–483

[3] Ng, A.Y., Russell, S.J.: 'Algorithms for inverse reinforcement learning'. Proc. of Int. Conf. on Machine Learning, Stanford, CA, USA, 2000, pp. 663–670

[4] Mahler, J., Pokorny, F.T., Hou, B., *et al.*: 'Dex-Net 1.0: a cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards'. Proc. of Int. Conf. on Robotics and Automation, Stockholm, Sweden, 2016, pp. 1957–1964

[5] Mahler, J., Liang, J., Niyaz, S., *et al.*: 'Dex-Net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics'. Robotics: Science and Systems, Cambridge, MA, USA, 2017

[6] Kober, J., Bagnell, J.A., Peters, J.: 'Reinforcement learning in robotics: a survey', *Int. J. Rob. Res.*, 2013, **32**, (11), pp. 1238–1274

[7] Escande, A., Mansard, N., Wieber, P.B.: 'Hierarchical quadratic programming: fast online humanoid-robot motion generation', *Int. J. Rob. Res.*, 2014, **33**, (7), pp. 1006–1028

[8] Zolkiewski, S., Pioskowik, D.: 'Robot control and online programming by human gestures using a kinect motion sensor', *New Perspect. Inf. Syst. Technol.*, 2014, **1**, pp. 593–604

[9] Zhu, G., Zhang, L., Shen, P., *et al.*: 'An online continuous human action recognition algorithm based on the kinect sensor', *Sensors*, 2016, **16**, (2), p. 161

[10] Golz, J., Wruetz, T., Eickmann, D., *et al.*: 'RoBO-2L, a Matlab interface for extended offline programming of KUKA industrial robots'. Proc. of Int. Conf. on Research and Education in Mechatronics, Campiegne, France, 2016, pp. 064–067

[11] Cai, Y., Yuan, P., Shi, Z., *et al.*: 'Application of universal kriging for calibrating offline-programming industrial robots', *J. Intell. Robot. Syst.*, 2018, pp. 1–10, doi: 10.1007/s10846-018-0823-7

[12] Ferreira, L.A., Figueira, Y.L., Iglesias, I.F., *et al.*: 'Offline CAD-based robot programming and welding parametrization of a flexible and adaptive robotic cell using enriched CAD/CAM system for shipbuilding', *Procedia Manuf.*, 2017, **11**, pp. 215–223

[13] Erdős, G., Kardos, C., Kemény, Z., *et al.*: 'Process planning and offline programming for robotic remote laser welding systems', *Int. J. Comput. Integr. Manuf.*, 2016, **29**, (12), pp. 1287–1306

[14] Pan, Z., Polden, J., Larkin, N., *et al.*: 'Recent progress on programming methods for industrial robots', *Robot. Comput. Integr. Manuf.*, 2012, **28**, (2), pp. 87–94

[15] Biggs, G., MacDonald, B.: 'A survey of robot programming systems'. Proc. of the Australasian Conf. on Robotics and Automation, Brisbane, Australia, 2003

[16] Schulman, J., Ho, J., Lee, A., *et al.*: 'Finding locally optimal, collision-free trajectories with sequential convex optimization'. Proc. of Robotics: Science and Systems, Berlin, Germany, 2013

[17] Kalakrishnan, M., Chitta, S., Theodorou, E., *et al.*: 'STOMP: stochastic trajectory optimization for motion planning'. Proc. of Int. Conf. on Robotics and Automation, Shanghai, China, 2011, pp. 4569–4574
[18] Siméon, T., Laumond, J.P., Cortés, J., *et al.*: 'Manipulation planning with probabilistic roadmaps', *Int. J. Rob. Res.*, 2004, **23**, (7–8), pp. 729–746
[19] Kuffner, J.J., LaValle, S.M.: 'RRT-connect: an efficient approach to single-query path planning'. Proc. of Int. Conf. on Robotics and Automation, San Francisco, CA, USA, 2000, pp. 995–1001
[20] Bicchi, A.: 'Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity', *IEEE Trans. Robot. Autom.*, 2000, **16**, (6), pp. 652–662
[21] Wolniakowski, A., Jorgensen, J.A., Miatliuk, K., *et al.*: 'Task and context sensitive optimization of gripper design using dynamic grasp simulation'. Proc. of Int. Conf. on Methods and Models in Automation and Robotics, Miedzyzdroje, Polan, 2015, pp. 29–34
[22] Liu, C., Qiao, H., Su, J., *et al.*: 'Vision-based 3-D grasping of 3-D objects with a simple 2-D gripper', *IEEE Trans. Syst. Man Cybern. Syst.*, 2014, **44**, (5), pp. 605–620
[23] Tournassoud, P., Lozano-Pérez, T., Mazer, E.: 'Regrasping'. Proceeding of Int. Conf. on Robotics and Automation, Raleigh, NC, USA, 1987, vol. 4, pp. 1924–1928
[24] Wan, W., Harada, K.: 'Reorientating objects with a gripping hand and a table surface'. Int. Conf. on Humanoid Robots, Seoul, South Korea, 2015, pp. 101–106
[25] Calandra, R., Owens, A., Jayaraman, D., *et al.*: 'More than a feeling: learning to grasp and regrasp using vision and touch', *IEEE Robot. Autom. Lett.*, 2018, **3**, (4), pp. 3300–3307
[26] Lynch, K.M., Mason, M.T.: 'Stable pushing: mechanics, controllability, and planning', *Int. J. Rob. Res.*, 1996, **15**, (6), pp. 533–556
[27] Odhner, L.U., Jentoft, L.P., Claffee, M.R., *et al.*: 'A compliant, underactuated hand for robust manipulation', *Int. J. Rob. Res.*, 2014, **33**, (5), pp. 736–752
[28] Odhner, L.U., Ma, R.R., Dollar, A.M.: 'Precision grasping and manipulation of small objects from flat surfaces using underactuated fingers'. Proc. of Int. Conf. on Robotics and Automation, St. Paul, MN, USA, 2012, pp. 2830–2835
[29] Dafle, N.C., Rodriguez, A., Paolini, R., *et al.*: 'Extrinsic dexterity: in-hand manipulation with external forces'. Proc. of Int. Conf. on Robotics and Automation, Hong Kong, China, 2014, pp. 1578–1585
[30] Chavan-Dafle, N., Holladay, R., Rodriguez, A.: 'In-hand manipulation via motion cones', *Dynamics*, 2018, **19**, (31), p. 18
[31] Karayiannidis, Y., Pauwels, K., Smith, C., *et al.*: 'In-hand manipulation using gravity and controlled slip'. Int. Conf. on Intelligent Robots and Systems, Hamburg, Germany, 2015, pp. 5636–5641
[32] Wan, W., Harada, K.: 'Regrasp planning using 10,000 s of grasps'. Proc. of Int. Conf. on Intelligent Robots and Systems, Vancouver, Canada, 2017, pp. 1929–1936
[33] David, P., DeMenthon, D., Duraiswami, R., *et al.*: 'Simultaneous pose and correspondence determination using line features'. Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Madison, WI, USA, 2003, vol. 2, pp. 424–431
[34] Jeong, W.Y., Lee, K.M.: 'Visual SLAM with line and corner features'. Proc. of Int. Conf. on Intelligent Robots and Systems, Daejeon, Korea, 2006, pp. 2570–2575
[35] Collet, A., Berenson, D., Srinivasa, S.S., *et al.*: 'Object recognition and full pose registration from a single image for robotic manipulation'. Proc. of Int. Conf. on Robotics and Automation, Kobe, Japan, 2009, pp. 48–55
[36] Moeslund, T.B., Kirkegaard, J.: 'Pose Estimation Using Structured Light and Harmonic Shape Contexts', in Braz, J., Ranchordas, A., Araujo, H., Jorge, J. (Eds.): Advances in Computer Graphics and Computer Vision, 2017, pp. 281–292
[37] Kirkegaard, J.: 'Pose estimation of randomly organized stator housings using structured light and harmonic shape contexts', Department of Health Science and Technology, Aalborg University, 2005
[38] Rusu, R.B., Bradski, G., Thibaux, R., *et al.*: 'Fast 3D recognition and pose using the viewpoint feature histogram'. Proc. of Int. Conf. on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 2155–2162
[39] Choi, S.M., Lim, E.G., Cho, J.I., *et al.*: 'Stereo vision system and stereo vision processing method'. US Patent 8,208,716, 2012
[40] Salvi, J., Pages, J., Batlle, J.: 'Pattern codification strategies in structured light systems', *Pattern Recognit.*, 2004, **37**, (4), pp. 827–849
[41] Avola, D., Cinque, L., Foresti, G.L., *et al.*: 'A practical framework for the development of augmented reality applications by using ArUco markers'. Proc. of Int. Conf. on Pattern Recognition Applications and Methods, Rome, Italy, 2016, pp. 645–654
[42] Koeda, M., Yano, D., Shintaku, N., *et al.*: 'Development of wireless surgical knife attachment with proximity indicators using ArUco marker'. Proc. of Int. Conf. on Human-Computer Interaction, Las Vegas, NV, USA, 2018, pp. 14–26
[43] Mihalyi, R.G., Pathak, K., Vaskevicius, N., *et al.*: 'Robust 3D object modeling with a low-cost RGBD-sensor and AR-markers for applications with untrained end-users', *Robot. Auton. Syst.*, 2015, **66**, pp. 1–17
[44] Yang, G., Saniie, J.: 'Indoor navigation for visually impaired using AR markers'. Proc. of Int. Conf. on Electro Information Technology, Lincoln, NE, USA, 2017, pp. 1–5
[45] Carmigniani, J., Furht, B., Anisetti, M., *et al.*: 'Augmented reality technologies, systems and applications', *Multimedia Tools Appl.*, 2011, **51**, (1), pp. 341–377
[46] Williams, B., Cummins, M., Neira, J., *et al.*: 'A comparison of loop closing techniques in monocular SLAM', *Robot. Auton. Syst.*, 2009, **57**, (12), pp. 1188–1197
[47] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., *et al.*: 'Automatic generation and detection of highly reliable fiducial markers under occlusion', *Pattern Recognit.*, 2014, **47**, (6), pp. 2280–2292
[48] Maier, D.: 'The theory of relational databases' (Computer Science Press, USA, 1983)
[49] Lozano-Pérez, T., Wesley, M.A.: 'An algorithm for planning collision-free paths among polyhedral obstacles', *Commun. Assoc. Comput. Mach.*, 1979, **22**, (10), pp. 560–570
[50] Ueda, J., Kondo, M., Ogasawara, T.: 'The multifingered NAIST hand system for robot in-hand manipulation', *Mech. Mach. Theory*, 2010, **45**, (2), pp. 224–238
[51] Dafle, N.C., Rodriguez, A., Paolini, R., *et al.*: 'Regrasping objects using extrinsic dexterity'. Proc. of Int. Conf. on Robotics and Automation, Hong Kong, China, 2014, pp. 2560–2560
[52] Cao, C., Wan, W., Pan, J., *et al.*: 'An empirical comparison among the effect of different supports in sequential robotic manipulation'. Proc. of Int. Conf. on Intelligent Robots and Systems, Daejeon, Korea, 2016, pp. 2548–2553
[53] Nguyen, V.D.: 'Constructing force-closure grasps', *Int. J. Rob. Res.*, 1988, **7**, (3), pp. 3–16
[54] Svestka, P., Latombe, J., Kavraki, L.O.: 'Probabilistic roadmaps for path planning in high-dimensional configuration spaces', *IEEE Trans. Robot. Autom.*, 1996, **12**, (4), pp. 566–580
[55] LaValle, S.M.: 'Rapidly-exploring random trees: a new tool for path planning'. Report No. TR98-11, Computer Science Department, Iowa State University, 1998, Available at http://janowiec.cs.iastate.edu/papers/rrt.ps
[56] Yershova, A., Jaillet, L., Simeon, T., *et al.*: 'Dynamic-domain RRTs: efficient exploration by controlling the sampling domain'. Proc. of Int. Conf. on Robotics and Automation, Barcelona, Spain, 2005, pp. 3856–3861
[57] Luders, B.D., Karaman, S., How, J.P.: 'Robust sampling-based motion planning with asymptotic optimality guarantees'. Proc. of AIAA Guidance, Navigation, and Control (GNC) Conf., Boston, MA, USA, 2013, p. 5097
[58] Jaillet, L., Cortés, J., Siméon, T.: 'Transition-based RRT for path planning in continuous cost spaces'. Proc. of Int. Conf. on Intelligent Robots and Systems, Nice, France, 2008, pp. 2145–2150
[59] Liu, H., Ding, D., Wan, W., *et al.*: 'Predictive model for path planning by using K-near dynamic bridge builder and Inner Parzen Window'. Proc. of Int. Conf. on Intelligent Robots and Systems, Nice, France, 2008, pp. 2133–2138
[60] Constantinescu, D., Croft, E.A.: 'Smooth and time-optimal trajectory planning for industrial manipulators along specified paths', *J. Robot. Syst.*, 2000, **17**, (5), pp. 233–249
[61] Berglund, T., Brodnik, A., Jonsson, H., *et al.*: 'Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles', *IEEE Trans. Autom. Sci. Eng.*, 2010, **7**, (1), pp. 167–172
[62] Yamanobe, N., Wan, W., Ramirez-Alpizar, I.G., *et al.*: 'A brief review of affordance in robotic manipulation research', *Adv. Robot.*, 2017, **31**, (19–20), pp. 1086–1101
[63] Hanai, R., Suzuki, H., Nakabo, Y., *et al.*: 'Modeling development process of skill-based system for human-like manipulation robot', *Adv. Robot.*, 2016, **30**, (10), pp. 676–690